PROJECT ABSTRACT

Speech Command Recognition using CNN (Google Speech Commands Dataset)

A built-to-order final-year project: a small-footprint CNN that recognises 35 spoken commands from one-second audio clips, trained on the Google Speech Commands dataset (105,829 utterances), with an audio console demo, TFLite export, report, PPT and viva kit.

01. Title

Speech Command Recognition using CNN (Google Speech Commands Dataset)

02. Introduction / Background

Voice interfaces depend on keyword spotting: a tiny model that hears 'stop' or 'go' and reacts in milliseconds on a microcontroller-class compute budget. The problem is genuinely hard — one second of 16 kHz audio is 16,000 samples carrying speaker variation, room noise and microphone colouration — and the model must stay small enough for the edge. Industry solves it by converting audio to log-Mel spectrograms and classifying them with compact convolutional networks. This project follows that exact recipe on the standard public benchmark, making it a strong final-year topic spanning signal processing and deep learning.

03. Problem Statement

Students need a project that goes beyond classifying clean images: real-world audio is noisy, variable and unforgiving, and keyword spotting adds a hard size/latency budget. The challenges are (1) representing speech in a learnable form, (2) generalising across 2,618 speakers and recording conditions, (3) surviving phonetically confusable pairs like no/go and three/tree, and (4) keeping the model small enough for on-device inference. This project addresses each with standard, defensible engineering.

04. Objectives

To train a compact CNN that recognises 35 spoken commands from one-second clips; to use the dataset's official splits for comparable evaluation; to apply the standard keyword-spotting augmentation recipe; to export the model to TensorFlow Lite with a documented size/latency profile; and to visualise the full pipeline in an audio console demo.

- 35-way classification from 40 × 98 log-Mel spectrograms.
- Design targets: ≈90% top-1 (35-way), ≈95% (12-command core subset).
- TFLite export under ~1 MB quantised with sub-15 ms CPU inference (expected).
- Confusion-pair analysis documenting exactly where errors concentrate.

05. Proposed Methodology

Each one-second 16 kHz clip is converted to a 40-band log-Mel spectrogram (30 ms window, 10 ms hop → 40 × 98 input). Training augmentation: background-noise mixing from the dataset's own

noise clips, ±100 ms time shifts and SpecAugment-style time/frequency masking. Model: three conv blocks (64/64/32 filters, batch-norm, max-pool, dropout) → global average pooling → dense(128) → softmax(35), ~200k parameters, trained with Adam and categorical cross-entropy. Evaluation uses the dataset's official SHA-1 train/validation/test lists; the test split is evaluated once. Quoted accuracies are design targets; the shipped report records the measured values.

06. System Architecture / Working

The pipeline has six stages. (1) Capture: microphone recording or sample utterance in the console, or dataset clips in training. (2) Feature extraction: log-Mel spectrogram computation, identical code in training and inference. (3) Augmentation (training only): noise mixing, time shifts, SpecAugment masking. (4) CNN: three convolution blocks extract time-frequency patterns, global average pooling compresses them, dense layers produce the 35-way softmax. (5) Export: the trained model is converted to TFLite and profiled. (6) Console: waveform, spectrogram heatmap and top-5 predictions rendered per utterance.

07. Hardware / Software Requirements

Software only; optional edge deployment targets commodity boards.

- Python 3 with TensorFlow/Keras, librosa, NumPy, Matplotlib, scikit-learn.
- Jupyter Notebook for the training and evaluation workflow.
- A modern browser (microphone permission) for the audio console demo.
- Laptop CPU trains the compact model; a GPU shortens it. No special hardware required.

08. Expected Outcomes

A trained keyword-spotting CNN meeting its design targets, exported to TFLite with a documented size/latency profile, an audio console that visualises every pipeline stage, and complete viva documentation. All metric figures are design targets for the built-to-order training run, not measured claims.

- 35-way top-1 accuracy ≈ 90% on the official test split (design target).
- 12-command core-subset accuracy ≈ 95% (design target).
- TFLite model under ~1 MB quantised, under 15 ms inference on laptop CPU (expected).
- Per-class metrics and confusion-pair analysis (no/go, three/tree, up/off, four/forward).

09. Applications

Keyword spotting is the front door of every voice interface.

- Wake-word and command layers for smart-home and robotics projects.
- Hands-free control for accessibility tools and workshop equipment.
- Voice-gated IoT prototypes (the CNN pairs naturally with an MCU-class board).
- Teaching aid for speech features, CNNs on spectrograms and on-device ML.

10. Future Scope

- Streaming decoder for continuous speech instead of isolated words.
- Quantisation-aware training and deployment on Cortex-M class microcontrollers.

- Speaker-adaptive fine-tuning for accented or children's speech.
- Multi-keyword sequences ('go left', 'stop now') via sequence models.